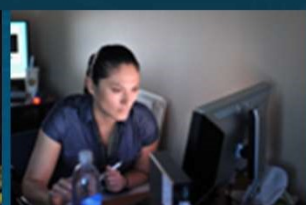
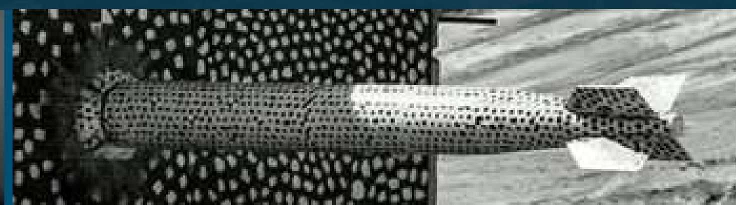
The IEEE International Conference on Rebooting Computing (ICRC 2020)

Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS





Tuesday, December 1st, 2020

Michael P. Frank, Center for Computing Research

with Robert W. Brocato, Brian D. Tierney, Nancy A. Missert, and Alexander H. Hsia

Approved for public release, SAND2020-12876 C

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline of Talk

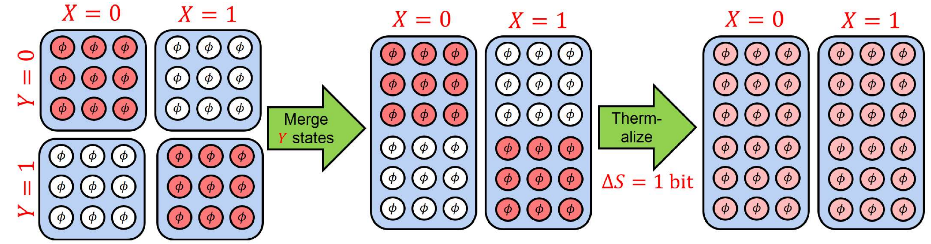
Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS

- I. Introduction: Motivation & Brief History.
- II. Requirements for Fully Static Adiabatic CMOS:
  - What is fully static operation, and why is it needed?
  - Limitations of some existing adiabatic CMOS families.
  - Requirements for fully static, fully adiabatic operation.
- III. Description of the S2LAL Logic Family:
  - Important notations; CMOS transmission gates.
  - Unlatched and latching static adiabatic buffers.
  - Reversible shift register structure and pipeline sequencing.
  - Logic gates and general logic functions
- IV. Future Work and Conclusion.

Section I. Motivation & History

Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS

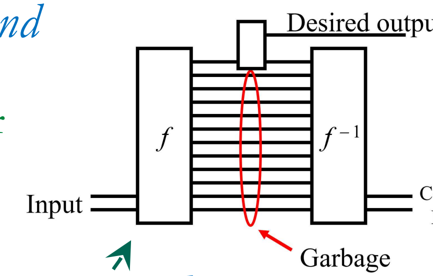
Motivation & Brief History



arXiv: 1901. 10327

Landauer's Principle (1961):

- Elementary statistical physics and information theory together imply that there is a *fundamental upper bound* on energy efficiency for the conventional (*non-reversible*) computing paradigm.
  - *Oblivious* erasure of known/correlated information implies dissipation of $E_{\text{diss}} \geq k_B T \ln 2$ energy to the environment for each bit's worth of known information that is lost.
    - k_B is Boltzmann's constant $\simeq 1.38 \times 10^{-2}$ J/K = the natural logarithmic unit of entropy.
    - **NOTE:** T is ***the temperature of the thermal environment into which the waste heat ends up getting ejected.***
      - ∴ Simply lowering T *locally* <u>cannot</u> help *directly* to lower *system-level* E_{diss} if the *external* environment temperature is fixed.

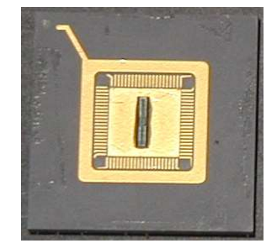


Reversible Computing (RC) provides a (theoretical, and eventually also practical!) solution:

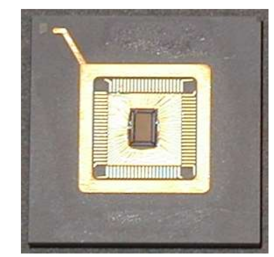
- RC means computing *without* oblivious erasure of known or correlated information.
  - In principle, energy dissipation per useful operation can be made *arbitrarily small* (can approach zero as technology improves).
    - ∴ Energy *efficiency* (operations per Joule) can theoretically approach *infinity* (or at least, no limits to this are yet known).
  - This includes implications for avoiding differential power analysis (DPA) and related side-channel attacks.

Some early history of the reversible computing field:

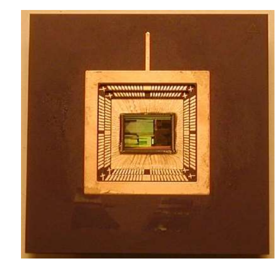
- RC was first shown *theoretically* coherent by Bennett, 1973 (doi:10.1147/rd.176.0525).
- First *engineering* implementation proposed by Likharev, 1977 (doi:10.1109/TMAG.1977.1059351).
- First fully-adiabatic sequential CMOS logic style: Younis & Knight, 1993 (Proc. Int'l Symp. Res. Int. Sys.).
- First fabricated reversible processor chips! Frank, Knight, Love, Margolus, Rixner, Vieri (1996-1999).



Tick



FlatTop



Pendulum

The time is ripe for a resurgence!

- I believe there is an opportunity right now to demonstrate some real breakthroughs.

Why Reversible Computing Wins Despite Its Overheads!

$$\eta = \frac{P}{C}$$

$$C_{\text{tot}} = C_{\text{mfg}} + C_{\text{oper}}$$

Bumper-sticker slogan: "***Running Faster by Running Slower!***" (Wait, what?) More precisely:

- Reversible technology is so energy-efficient that we can overcome its overheads (including longer transition times!) by using much greater parallelism to increase overall performance within system power constraints.

Bottom line: The computational *performance per unit budgetary cost* on parallelizable computing workloads can become as large as desired, given only that *both terms* in this expression for total *cost per operation* C_{op} can be made sufficiently small:

$$C_{\text{op}} = c_E \cdot E_{\text{diss,op}} + c_M (s_{\text{elem}} \cdot t_{\text{delay}}).$$

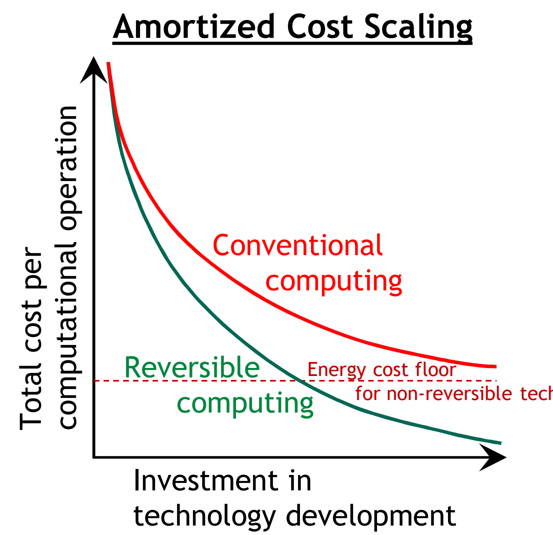
where:

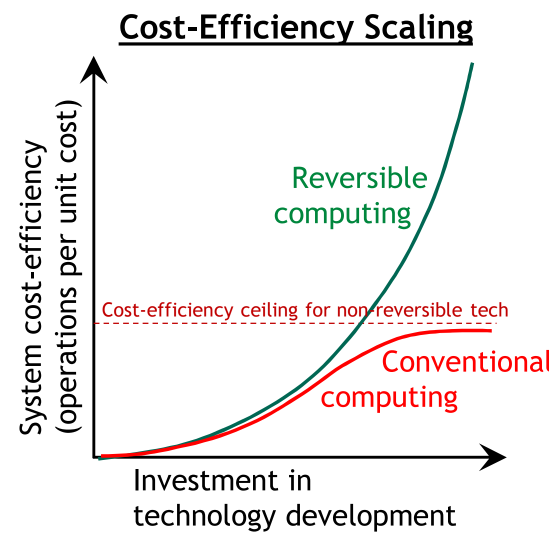
- c_E is the operating cost C_{oper} attributable to supplying power/cooling, divided by energy delivered.
- $E_{\text{diss,op}}$ is the system energy dissipation, divided by number of operations performed.
- c_M is the total cost C_{mfg} for system manufacturing & installation, *divided by* the physical size s_{elem} of individual computing elements (in appropriate units), & the system's total useful lifetime t_{life}.
- t_{delay} is the average time delay between instances of re-use of each individual computing element.

Two key observations:

- The cost per operation of *all* conventional computing *approaches a hard floor* due to Landauer.
  - Assuming *only* that the economic cost of operation *per Joule delivered* cannot become arbitrarily small.
- But, there is *no* clear barrier to making the manufacturing cost coefficient c_M *ever smaller* as manufacturing processes are refined, and/or the deployed lifetime of the system increases.

∴ *Nothing prevents* system-level cost efficiency of reversible machines from becoming *arbitrarily* larger than conventional ones, *even* if we have to scale t_{delay} and/or s_{elem} up as we scale $E_{\text{diss,op}}$ down!





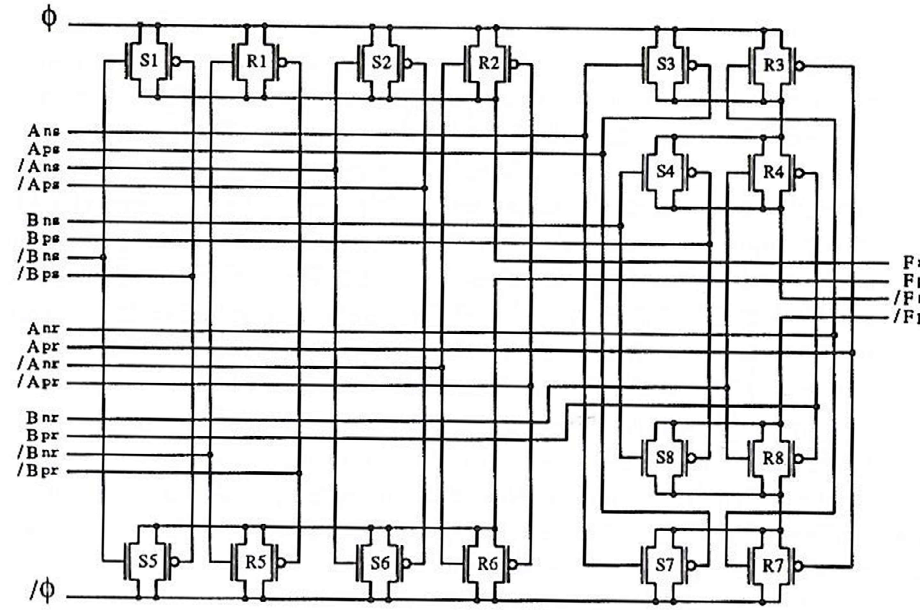
Early Examples of Fully Adiabatic CMOS Logic Families

S. G. Younis and T. F. Knight, Jr., “Practical implementation of charge recovering asymptotically zero power CMOS,” in Research on Integrated Systems: Proc. 1993 Symp., C. Ebeling and G. Borriello, Eds. Cambridge: MIT Press, Feb. 1993, pp. 234–250.

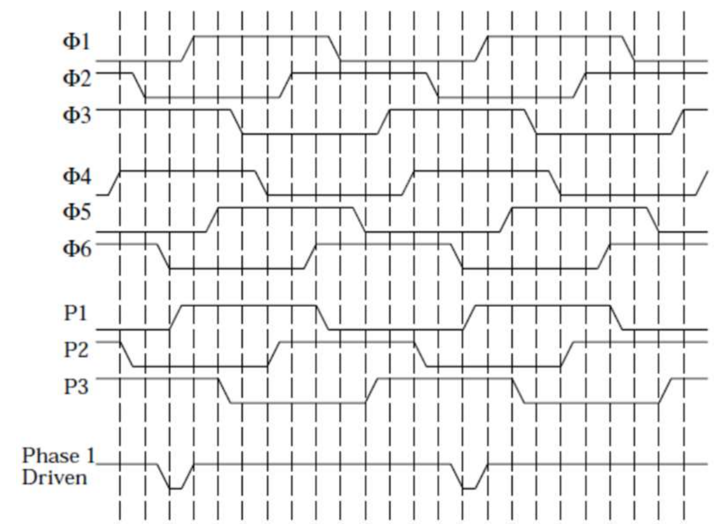
- First fully adiabatic, general sequential CMOS logic family.
- Four clock phases, four transitions per clock cycle.
- Quad-rail logic encoding.
- Slightly generalized by the 2LAL logic family (Frank, 2000).
- Dynamic logic.

S. G. Younis, “Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic,” Ph.D. thesis, Massachusetts Institute of Technology, June 1994. dspace.mit.edu/handle/1721.1/11620

- Simplified hardware designs compared to CRL.
  - Single-rail logic is possible.
- Several clocking variants, including “static” versions.
- Contains a minor non-adiabatic/non-static bug, I discovered in ‘97.
  - Easily fixed, however, by adding 1 extra transistor per logic gate.



Younis & Knight ‘93: CRL 2-input NAND gate.



Younis ‘94: Clocks for 24-tick “static” SCRL

Goal of This Work

Design a new sequential, pipelined adiabatic CMOS logic family with the following features:

1. ***Fully adiabatic operation.***
   - *I.e.*, no non-adiabatic "spark" or "squelch" events occur in an ideal setting.
     - *I.e.*, given negligible leakage and parasitic couplings.
2. ***Fully static operation.***
   - *I.e.*, each circuit node is connected to a supply at all times.
     - Note, this feature facilitates elimination of non-adiabatic events even in *non-ideal* settings.
3. ***Minimal latency.***
   - *I.e.*, only one "tick" or transition time of delay per layer of combinational logic depth.
4. ***Maximum throughput.***
   - *I.e.*, the number of ticks per initiation interval should also be minimal.
     - **Conjecture:** Minimum clock period meeting other design goals is 8 ticks (achieved in this work).

Section II. Requirements

Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS

Basic Requirements for Fully Adiabatic Operation

No diodes in charging paths!

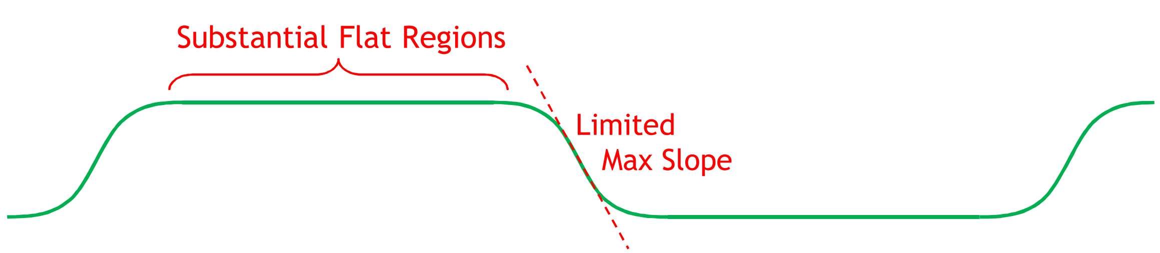
- All diodes have a built-in voltage drop for fundamental thermodynamic reasons.

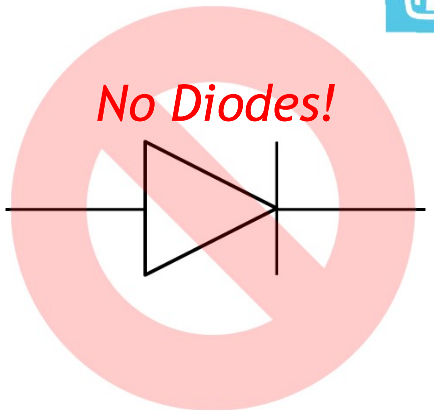
Operate all switches (*e.g.*, FETs) with a "dry-switching" discipline:

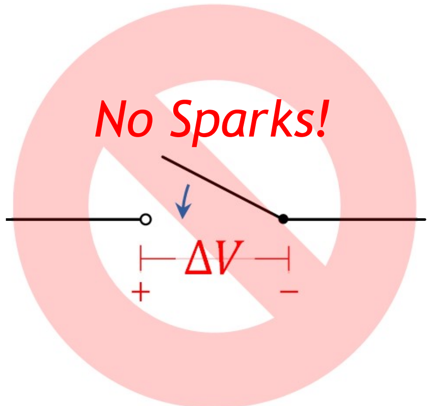
- Never turn on (close) a switch when there is a significant voltage difference $\Delta V \neq 0$ between its terminals.
  - Leads to a sudden, non-adiabatic flow of current.
  - More generally: No rapid voltage changes.
- Never turn off (open) a switch when there is a significant current flow $I \neq 0$ through the switch.
  - Leads to non-adiabatic losses as switch is (non-instantaneously) turning off.
    - Resistance through switch increases during turnoff → voltage drop increases → non-adiabatic loss across voltage drop.
  - Exception: If path is low inductance and there is an alternate path for the current.

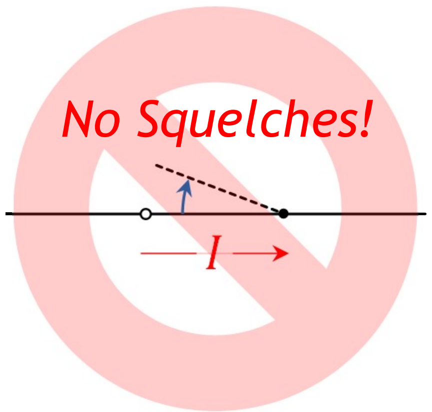
Use quasi-trapezoidal driving waveforms (no steep edges; flat tops and bottoms).

- This is necessary to obey the other rules.









Why Static Adiabatic Logic?

In non-static (*i.e.*, *dynamic*) logic styles, by definition, some circuit nodes are allowed to *float* dynamically (*i.e.*, without any direct tie to source) for at least part of the time.

- *E.g.*, this happens in a dynamic random-access memory (DRAM) cell.

The problem with having floating nodes is that their voltages may *vary from their ideal level* while they are isolated, for example, due to:

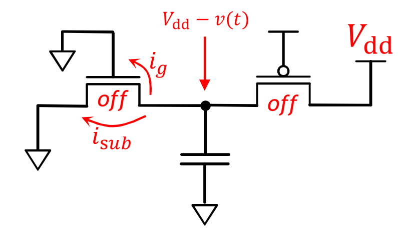
- *Voltage drift* due to leakage currents to sources at different levels through nominally turned-off devices. Includes:
  - Subthreshold leakage current $i_{sub}(t)$ across the channel of a device below threshold.
  - Gate leakage current $i_g(t)$ due to tunneling through the gate oxide.
- *Voltage sag* due to capacitive voltage-division effects involving parasitic capacitive couplings to nearby nodes with time-varying voltages.

If a floating node with capacitance C has a voltage disparity of ΔV from a given reference level at the time that it is reconnected to a source at that level,

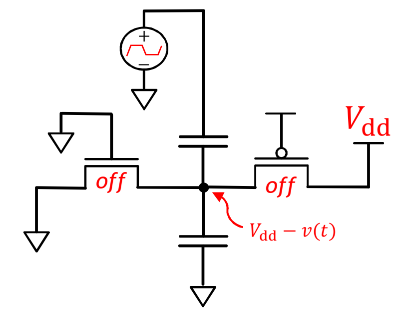
- Then there will be a sudden non-adiabatic "sparking" event dissipating $C(\Delta V)^2/2$ energy at the time of reconnection.

Avoiding these sparking events would require very precise engineering of all the possible paths for leakage and sag (*e.g.* to ensure the effects cancel)…

- OR, we could just design a fully static logic family! ← ***Much easier!***



Voltage drift due to leakage

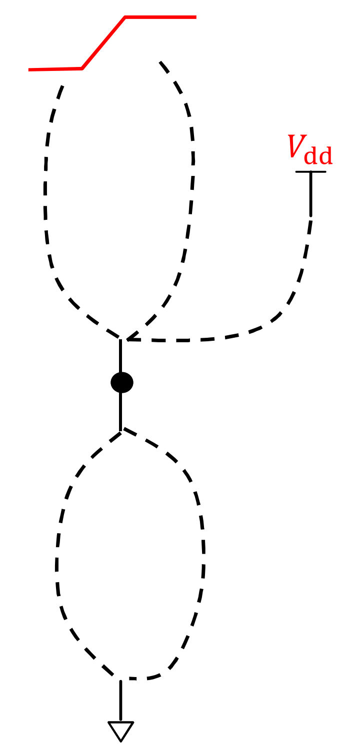


Voltage sag due to capacitive coupling to nearby varying nodes

Rules for Fully Static Operation

At *all times*, *each* internal node of the circuit must be connected to a voltage reference in one of the following manners:

1. Connected via a medium-impedance path through turned-on transistor(s) to a single constant-voltage reference;
2. Connected via a medium-impedance path through turned-on transistor(s) to a single variable-voltage reference;
3. Connected in a way that is actively transitioning (in either direction) between conditions 1 & 2 above,
   - with one path in the process of being connected while the other is in the process of being disconnected, and
   - where, at any given time throughout the transition, at least one path has no more than medium impedance, and
   - where, throughout the transition period, the level of the variable-voltage reference in question is being held constant at the same level as the constant-voltage rail;
4. Connected in a way that is (similarly) actively transitioning between two different paths to a single supply reference (whether it is constant-voltage or variable-voltage).



Where "medium impedance" means below some reasonable upper limit (*e.g.* 100 kΩ).

- And, all paths that are nominally "off" should have a much higher impedance, e.g., >>1 MΩ.

Section III. S2LAL Description

Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS

Notations and Conventions Used *(slide 1 of 2)*

Two nominal voltage levels: 0 V (GND, "low") and $V_{\mathrm{dd}} \gtrsim 2|V_{\mathrm{t}}|$ ("high").

Divide time into equal, discrete intervals called *ticks*, each of duration $\bar{\tau}_{\mathrm{tr}}$, and numbered consecutively.

- Every *transition* between nominal levels is required to fit entirely within a tick,
  - so, the actual transition time τ_{tr} is upper-bounded by the tick length, $\tau_{\mathrm{tr}} \leq \bar{\tau}_{\mathrm{tr}}$.

The active energy dissipation from any given adiabatic transition is as follows:

$$E_{\mathrm{a}} = \xi_{\mathrm{tr}} C_{\mathrm{L}} V_{\mathrm{dd}}^2 \frac{R C_{\mathrm{L}}}{\tau_{\mathrm{tr}}},$$

where:

- ξ_{tr} is a constant *shape factor* that accounts for the departure of the ramp shape from the ideal;
- C_{L} is the capacitive load of the node that is transitioning;
- R is the effective resistance of the charging path.

The clock period τ_{p} is an integer number n of ticks, $\tau_{\mathrm{p}} = n\bar{\tau}_{\mathrm{tr}}$.

- Thus, the clock frequency is

$$f = (n\bar{\tau}_{\mathrm{tr}})^{-1}.$$

- Ticks within a cycle are numbered modulo n (*i.e.*, $0, \ldots, n-1$).

Notations and Conventions Used *(slide 2 of 2)*

In the logic styles we'll discuss, any given logic *symbol* L (e.g., 0 or 1) is represented by a complementary *signal pair*.

- Thus, for k-valued logic we require $2k$ signals.
- Normally we have just $k = 2$ symbols, $L \in \{0,1\}$.

Possible conditions for a given signal pair (when valid) are *active* or *inactive*.

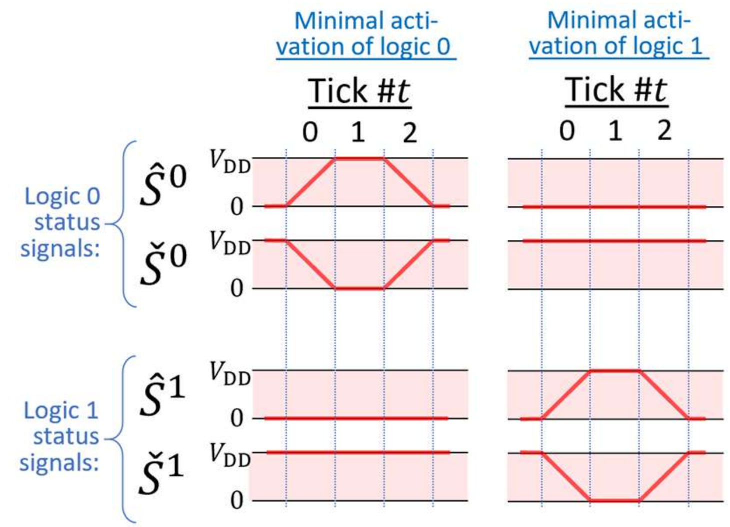
- One of the signals in each pair is *active-high*; the other is *active-low*.
  - When in the active state, we say the pair is *actively representing* the corresponding logic symbol L.
- The signal pair may feed the control terminals of a CMOS transmission gate.
  - The active-high signal controls the nFET, and the active-low signal controls the pFET.
  - Thus, the transmission gate is turned ON (conducting) when the signal pair is active.
  - The body terminals of the FETs should be separately biased (not tied to either channel terminal).
    - Can be used to increase device thresholds if desired.

The following notation is used for a signal pair:

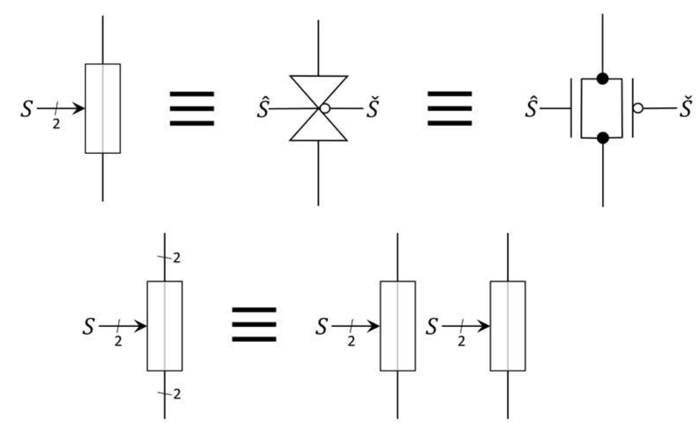
$$S^L_{t_\mathrm{b},t_\mathrm{e}} = (\hat{S}^L_{t_\mathrm{b},t_\mathrm{e}}, \check{S}^L_{t_\mathrm{b},t_\mathrm{e}})$$

where:

- $\hat{}$, $\check{}$ accents denote active-high and active-low signals, respectively.
  - No accent denotes the pair.
- L (if present) denotes the logic symbol the signal pair is representing.
- $t_\mathrm{b}, t_\mathrm{e}$ (if present) denote the transitional (*begin* and *end*) ticks of the active period.



Examples of minimal activations

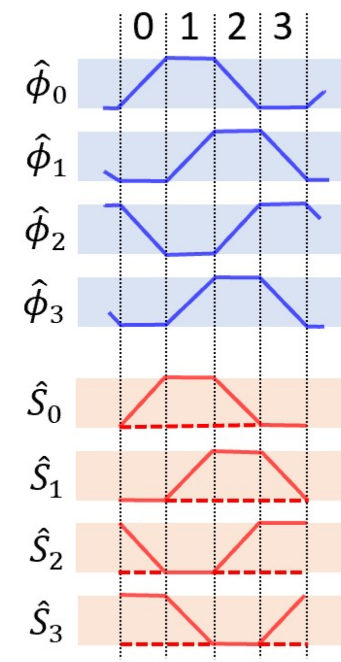


Transmission gate symbols

Review of 2LAL

2LAL is a simple variant of CRL, first described by M. Frank in lectures at the University of Florida in 2000.

- Four clock phases, each active for one tick and inactive for one tick.
- A simple (one-symbol) shift register structure is shown.
- Series/parallel combinations of transmission gates can be used to do logic (not shown here).
  - 2LAL really only differs from CRL in terms of allowing more flexibility in how internal nodes are handled

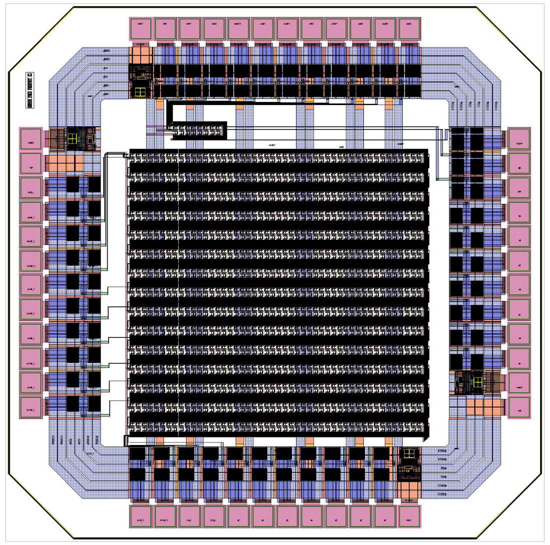


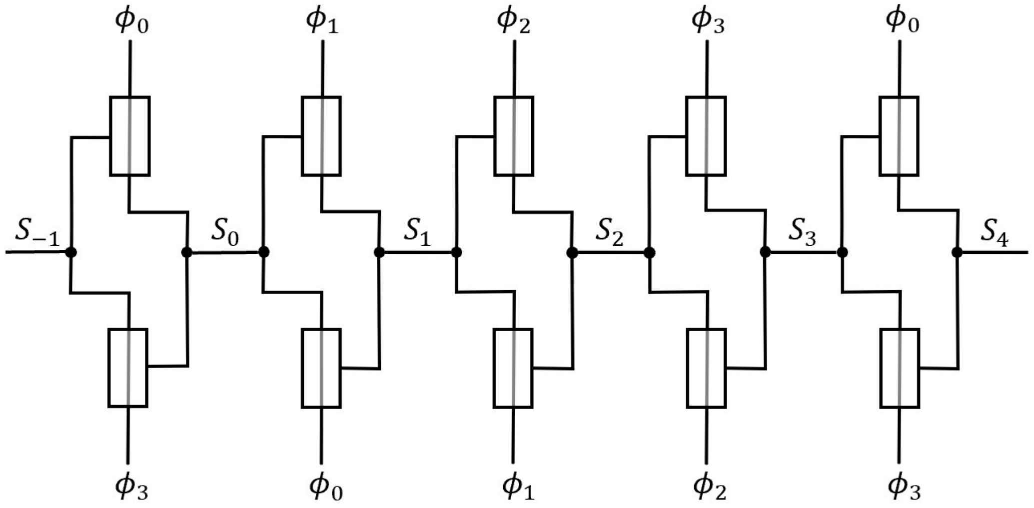
Simulation results for 2LAL obtained at Sandia in 2020:

- Energy dissipation per cycle per FET in shift register @50% activity factor at $f = 1$ MHz, $C_L = 10$ fF:
  - Spectre simulation of MESA 350 nm, $W = 800$ nm: **37 aJ ≈ 230 eV.**
  - Spectre simulation of MESA 180 nm, $W = 480$ nm: **6.9 aJ ≈ 43 eV.** ← Comparable to a data point for TSMC18 from 2004.
    - This beats end-of-roadmap standard CMOS substantially.

Test chip taped out in Aug. 2020:

- MESA 180 nm shuttle run.
- 2×2 mm die.
- 8-stage & 720-stage shift registers.
- Goal: Verify function & dissipation.

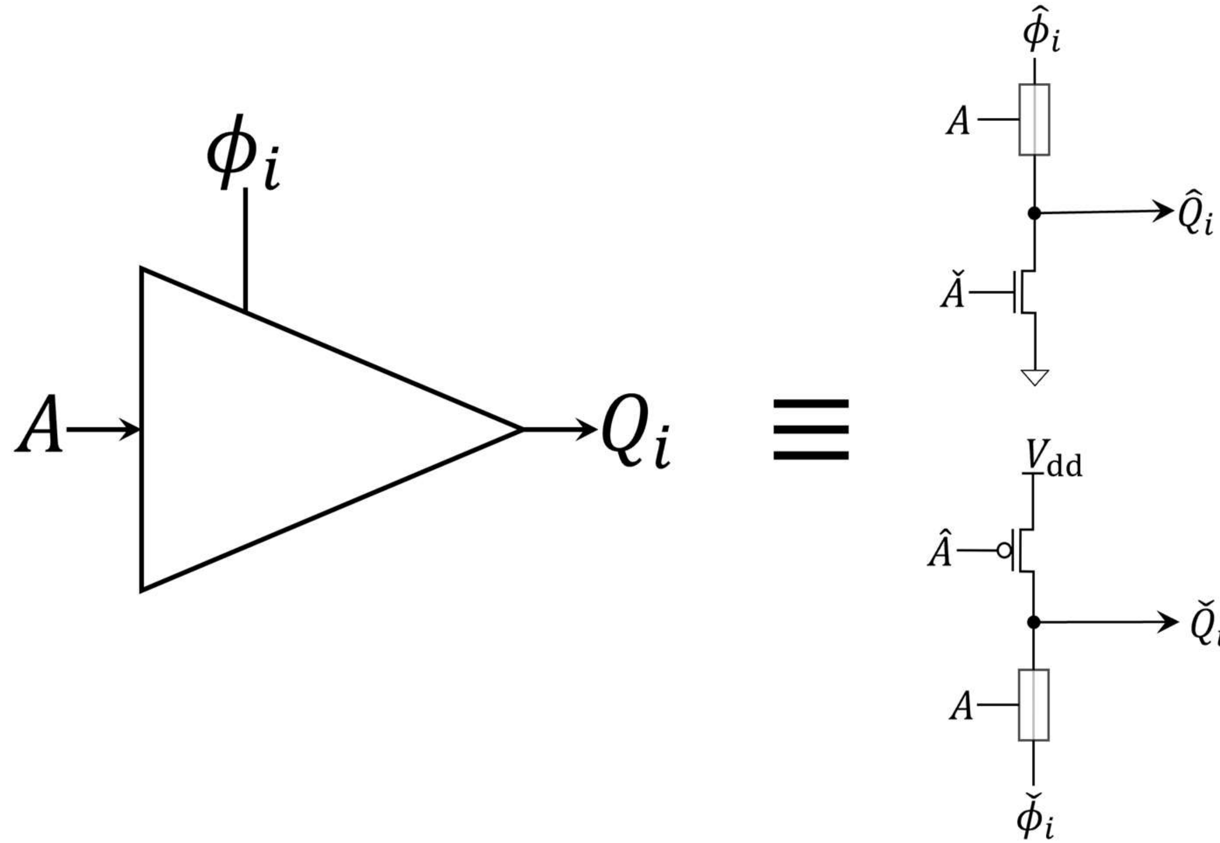


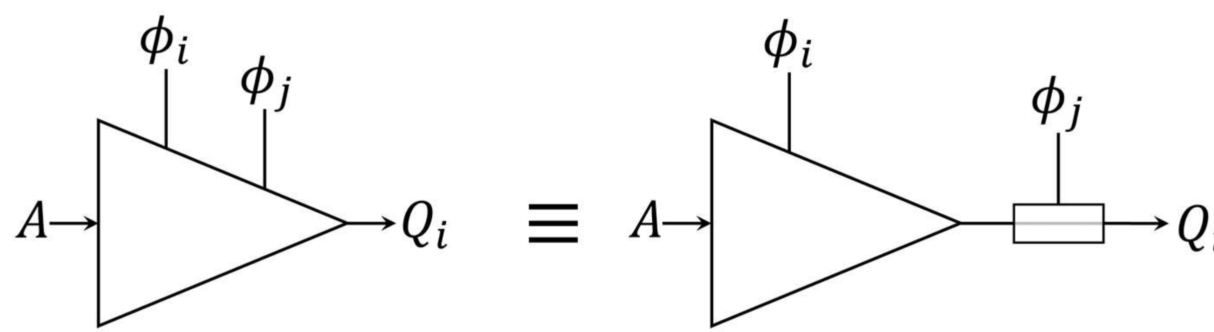


Basic Elements of S2LAL

Unlatched & Latching Static Adiabatic Buffers

- Unlatched version exchanges control of output between clock and fixed supply, depending on activity of input.
  - Handoff should only happen when levels match.
- Latching version uses an out-of-phase clock to latch (or unlatch!) the output.
  - NOTE: This requires additional structure to make it fully static!





S2LAL Reversible Pipeline Structure

Paired forward and reverse stages:

- Forward stages activate to compute *later* signals from *earlier* ones.
- Reverse stages *de-activate* to *de-compute earlier* signals from *later* ones.

Every signal S_i must stay active for (at least) 5 ticks:

- Provides sufficient time for the following sequence of steps:
  - (1) Activate forwards stage F_{i+1}, (2) Activate reverse stage R_i, (3) Handoff control of S_i from F_i to R_i, (4) Deactivate forwards stage F_i, (5) Deactivate reverse stage R_{i-1}.

Add 3 ticks for transitions & inactive handoff:

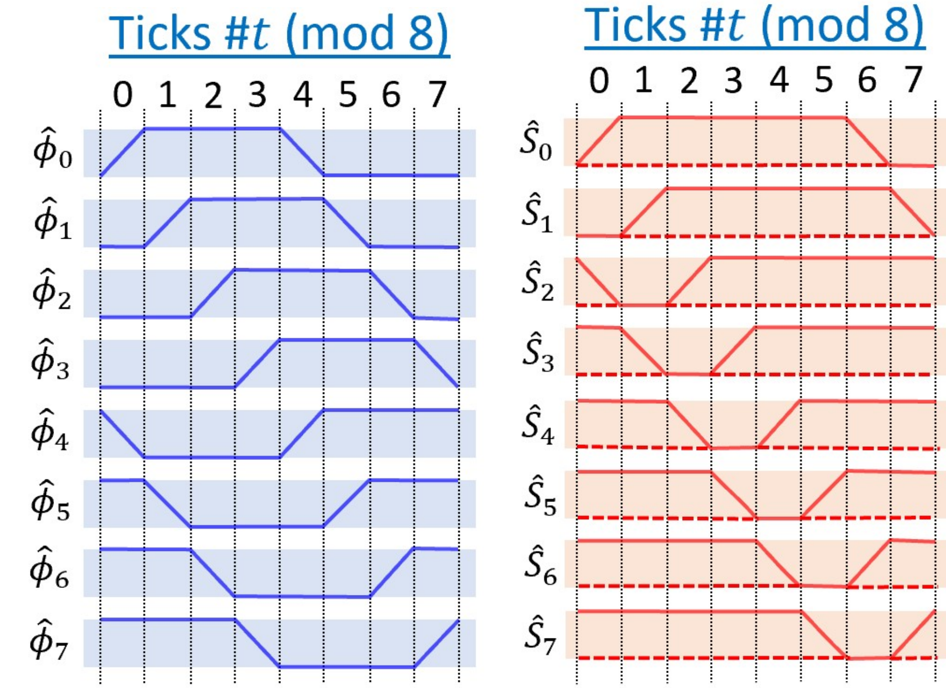
- Total cycle length = **8 ticks** min.

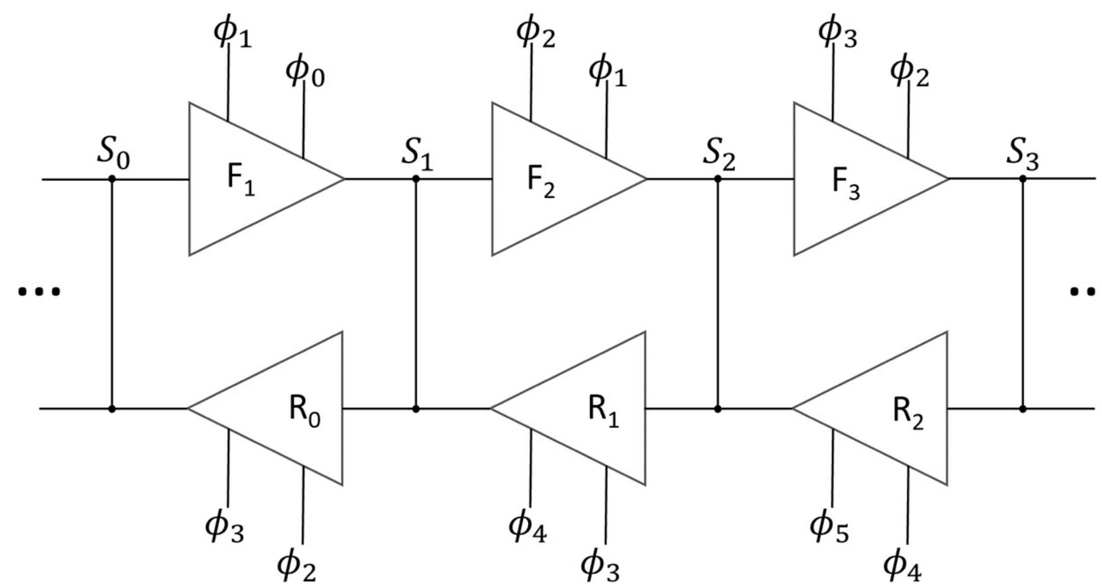
Note control of each signal S_i is handed off to forward stage F_i on ticks $\#i-1$, and to reverse stage R_i on ticks $\#i+3$.

- Signal S_i goes valid on ticks $\#i$ and invalid (inactive) on ticks $\#i+6$.

For general logic, functions must be invertible.

- Optimizing whole pipeline gets into reversible algorithm design: Considered out of scope for this particular paper.





S2LAL Logic Gates

14-transistor AND gate, 16-transistor OR gate.

- Carefully designed to ensure that each internal node is always connected to either constant or variable source.
  - The structures shown are minimal, given the design constraints.

Inverting gates are done easily, by using signal pairs for complementary symbols:

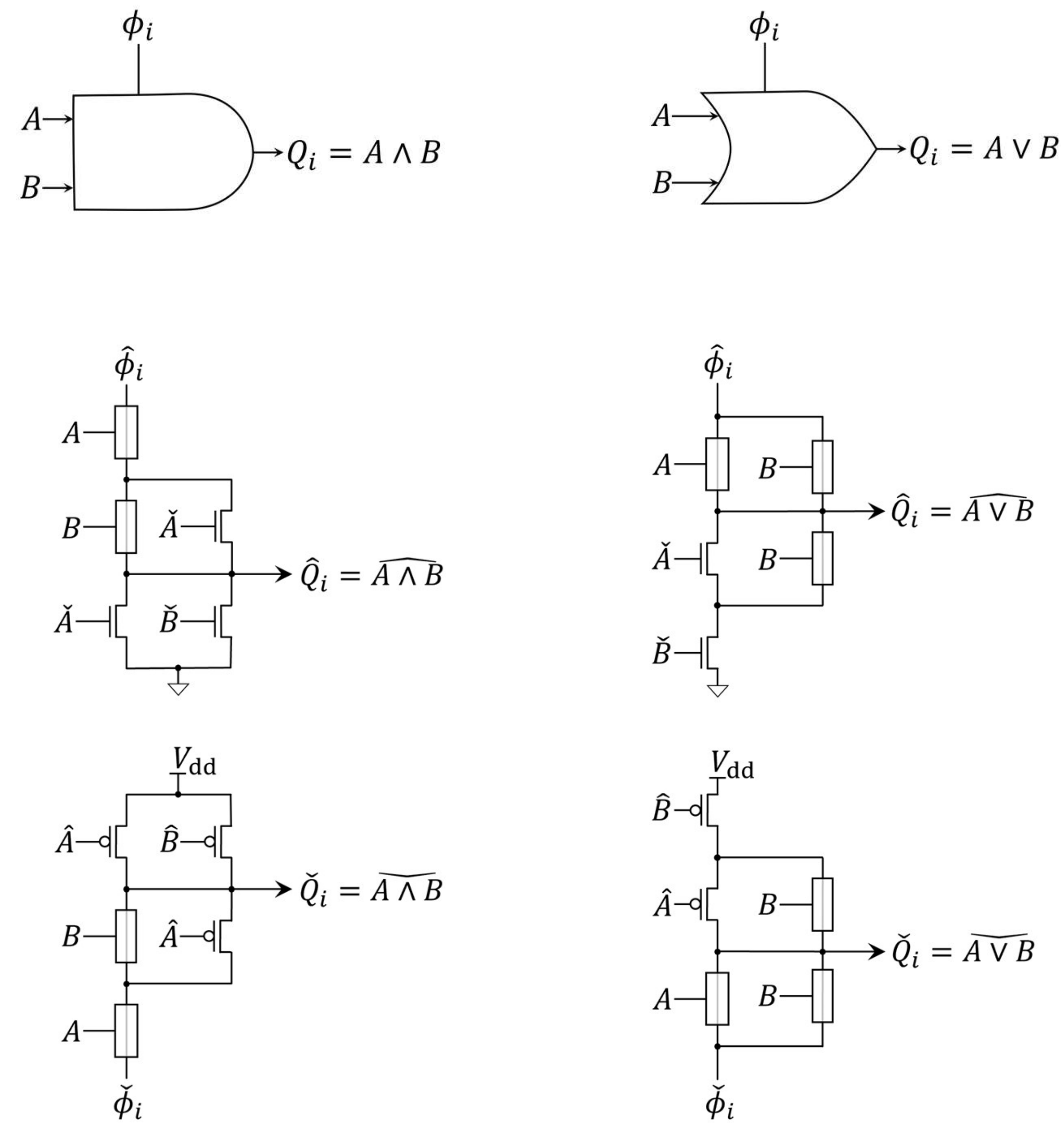
- $\mathrm{NOT}(A^1) = \mathrm{BUFFER}(A^0)$
- $\mathrm{NAND}(A^1, \mathrm{B}^1) = \mathrm{OR}(A^0, B^0)$
- $\mathrm{NOR}(A^1, \mathrm{B}^1) = \mathrm{AND}(A^0, B^0)$

Also! Erik DeBenedictis invented an optimization to S2LAL that can compute the inverses as-needed, rather than keeping both the 0,1 signal pairs around:

- See https://zettaflops.org/zf004/ .

AND

OR



Section IV. Future Work & Conclusion

Reversible Computing with Fast, Fully Static, Fully Adiabatic CMOS

Future Work

Some next steps:

1. Simulation studies.
   - Expect the minimum dissipation in realistic simulations to be lower than that of 2LAL.
2. Fabrication & power dissipation measurement of S2LAL test chips.
   - Validate simulation results.
3. Open-source hardware.
   - Open-source library of reference cells and example designs for static adiabatic CMOS.
   - Target an open PDK? (Sky130?)
4. Cryogenic technologies.
   - ***Ultra-low dissipation.*** Steeper subthreshold slope, lower off-state current, re-optimize device structure to reduce gate leakage also.
   - ***Power supply decoupling.*** For cryo applications, can move the supply to the room-temperature environment.
   - ***Superconducting interconnects.*** Improves the energy-delay product due to reduced parasitic resistance.
5. High-Q resonant supplies.
   - Currently under development at Sandia. (Provisional patent available under NDA.)
   - Superconducting versions are possible.

Conclusion

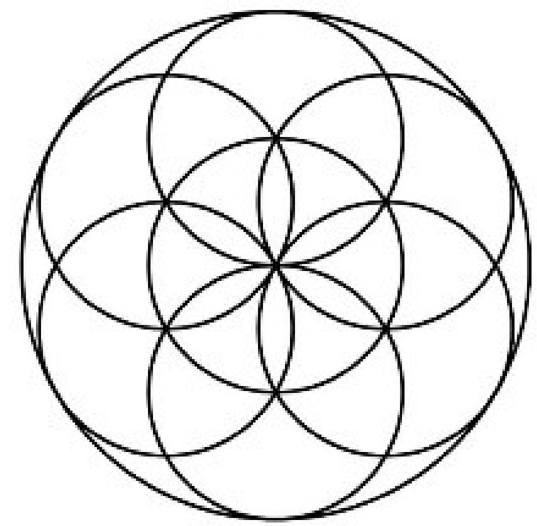
We have presented S2LAL, the first (and fastest!) form of fully static, fully adiabatic general sequential CMOS logic.

- We feel that these logic styles deserve the term "***perfectly adiabatic***."

In principle, given a sufficiently low-leakage process, S2LAL should be capable of ***outperforming the energy efficiency of any other known semiconductor-based form of digital logic***.

- S2LAL exhibits significant potential for both record-breaking scientific demonstrations, as well as relatively near-term practical applications, particularly:
  - Applications in cryogenic environments.
  - Any computing applications where energy efficiency is at a premium.

Also, S2LAL illustrates that ***vast gains in efficiency can still be achieved for general digital computing, even in CMOS***, but *only* if we take the principles of reversible computing seriously, and develop implementations of them with care!



Acknowledgement:
Thanks to Mr. Richard Magnano for helping to inspire this innovation.